OIPE
MAR 2 4 2004
JC98
TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

RECEIVED
MAR 2 6 2004
Technology Center 2100

Appl. No.	:	09/496,844
Applicant	:	Patrick KNEBEL, et al.
Filed	:	February 2, 2000
Title	:	METHOD AND COMPUTER SYSTEM FOR DECOMPOSING MACROINSTRUCTIONS INTO MICROINSTRUCTIONS AND FORCING THE PARALLEL ISSUE OF AT LEAST TWO MICROINSTRUCTIONS (Amended)
TC/A.U.	:	2183
Examiner	:	Huisman, David J.
Docket No.	:	10971393-1
Customer No.	:	022879

Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

DECLARATION OF PATRICK KNEBEL PURSUANT TO 37 C.F.R. § 1.131

Dear Sir:

1. Prior to May 18, 1999, the filing date of U.S. Patent No. 6,330,657 to Col et al., we reduced to practice the invention described in the claims of the present application.

2. At least by February 24, 1999, we reduced to practice a method and computer system that include decomposing macroinstructions into microinstructions, forcing the parallel issue of at least two microinstructions, and if an exception occurs in any of the microinstructions, canceling all of the microinstructions.

3. Attached as Exhibits 1 and 2 are two sets of RTL codes that evidence the reduction to practice of these features. The RTL code in Exhibit 1 is dated February 24, 1999. The RTL code in Exhibit 2 is dated December 4, 1998.

4. The RTL code in Exhibit 1 illustrates forcing the parallel issue of at least two microinstructions. The RTL code in Exhibit 2 illustrates if an exception occurs in any of the microinstructions, canceling all of the microinstructions.

[illegible] 3/24/04

5. The acts related above all took place in the United States of America.

6. The declarant further states that the above statements were made with the knowledge that willful false statements and the like are punishable by fine and/or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

Date: March 24, 2004

[illegible]

Patrick Knebel

WAS:104041.1

MAR 2 4 2004
TRADEMA

RECEIVED
MAR 2 6 2004
Technology Center 2100

```
!=====================================================================
!&! $Header: /h1/logic/rtl/dev/hdl/src/bnddeps.hdl 1.25 1999/02/24 03:51:54 h1cad
Exp $
!=====================================================================
!
! bnddeps.hdl
!
! Author:           Patrick Knebel
! Date:             10/22/97
! Copyright (c) 1997, 1998, 1999 Hewlett Packard Corporation
!
!------------------------------ Contents ------------------------------
!
! Bifrost Bundle Builder; The BuNDler!
! Takes the uops and immediates from UDQ, checks them for dependencies,
! and ships syllables off to IFR's Instruction Buffer.
!
!
!        -------------------------
!        | DE4 | SCH | XMT | ROT |
!        -------------------------
!           |     |     |     |
!           |     |     |     +---- read from IB, rotate, send to dispersal
!           |     |     +---------- convert to EM bundles, send to EM, write IB
!           |     +---------------- check syllable dependencies
!           +---------------------- read UDQ
!
! bnddeps.hdl
! Control block for checking whether to issue 1 or 2 syllables to EM.
! Check register dependencies and other control bits to determine dual issue.
!
!----- Modification History ---- Also see fhist(1) for an SBCM log ----------
! 990218 anuj- add clock gaters for powerup/down bifrost clocks
! 981211 rcb - bug fix, cond7 needed to include any non-zero subop type
! 981118 pjk - move register dependency checking from bnddeps to bndpaird
! 980715 pjk - remove smc
! 980501 pjk - change fp-stkptr dependency rules
! 971218 pjk - yet another (different) way to encode uop-subtype.  Instead of
!              using the 1-bit aflag field, create a new 3-bit subtype field
! 971111 pjk - put in new uoptype encodings
!              use aflags (future subuoptype) in place of ftosrd and store-op
! 971022 pjk - new
!

FUB bnddeps;

BEGIN  "bnddeps"

!------------------------------- Included Files ----------------------------

sourcefile "p7includes.def"
sourcefile "syn_maclib.def"
sourcefile "bif.def"
sourcefile "p7params.def"
sourcefile "uop.def"
sourcefile "logical_register.typ"
sourcefile "idu.def"

!------------------------ Ports, signals, and clock ----------------------
sourcefile "bnddeps.ifc"
sourcefile "bnddeps.int"
sourcefile "clock.typ"

!-------------------------- Compile Time Variables ------------------------
```

```
CtINTEGER j endc

!----------------------------- MAIN ---------------------------------------

STRUCTURAL MAIN;
BEGIN "bnddeps MAIN"

! Generate internal clock
GaterCkL(clk, BndClkEn111B, BnddepsGCInt, qclkbnddeps);

!-----------------------------------------------------------------------
! Bundle logic decides whether to send 0, 1, or 2 uops to EM
!-----------------------------------------------------------------------
! Hardware issues 0 uops if:
! There are no uops to send
bndnop1 := vNOT(BndUopMuxUopValid_0SchB);

! or if older uop is "bundle always with successor" and there is no successor (yet)
bndnop2 := vAND3(vNOT(BndUopMuxBndHint_0SchB[1]),
                 BndUopMuxBndHint_0SchB[0],
                 vNOT(BndUopMuxUopValid_1SchB));

BndNoIssueSchB := vOR2(bndnop1, bndnop2);

! Note: BndNoIssueSchB has priority over BndDualIssueSchB


! Hardware can dual-issue 2 uops iff:

! Diagnose bit to turn off bundling
! (fixme: short-term just hack it)
bndcond0 := NOT EmtSnglIssEnB;  ! enable bundling

! Older inst is not (sync or never pair with successor)
! Ynger inst is not (sync or always pair with successor)
bndcond1 :=  vNOR2(BndUopMuxBndHint_0SchB[1],
                   BndUopMuxBndHint_1SchB[0]);

! 2nd uop is not i0
! i0 must not bundle with predecessor
bndcond2 := vNAND3(BndUopMuxUopType_1SchB[2],
                   BndUopMuxUopType_1SchB[1],
                   vNOT(BndUopMuxUopType_1SchB[0]));

! 2 or more uops available from udq and the save latches
bndcond3 := BndUopMuxUopValid_1SchB;

! 1st and 2nd uops not both (load/store/bif) ops
! fixme: should we combine this rule and the fp-stk rule
!        to disable all combinations of bif/f/m0/m2?
! fixme: should we allow 2 loads/cycle?  TLL has some data
!        debug event muxes that would have to change.
bndcond4 := vNAND4(vNOT(BndUopMuxUopType_0SchB[2]),
                   vOR2(BndUopMuxUopType_0SchB[1],
                        vNOT(BndUopMuxUopType_0SchB[0])),
                   vNOT(BndUopMuxUopType_1SchB[2]),
                   vOR2(BndUopMuxUopType_1SchB[1],
                        vNOT(BndUopMuxUopType_1SchB[0])));

! no fp-tos WAW or RAW hazard
! don't bundle if both uops are Bif/f/m0/m2 and subtype is fp
bndcond5 := vNAND4(vNOT(BndUopMuxUopType_0SchB[2]),
                   BndUopMuxSubType_0SchB[2],
                    vNOT(BndUopMuxUopType_1SchB[2]),
                    BndUopMuxSubType_1SchB[2]);

! both uops are not flushing tbit.iA or cmp.iA
bndcond7 := vNAND4(BndUopMuxUopType_0SchB[2],
                   BndUopMuxSubType_0SchB <> '000,
                   BndUopMuxUopType_1SchB[2],
                   BndUopMuxSubType_1SchB <> '000);

! no eflags WAW hazard (RAW can't happen)
! This check is no longer needed.  The eflags h/w in the prodf unit
! can handle WAW hazards.  And RAW cannot happen because the only readers
! are tbitia and prodf which are both I0 syllables (always older).
! bndcondX := 1;

! the following rule is now covered by fp-tos hazard check above
! both uops are not of F type
! (ucode can override this by issueing "always bundle" hint)
!bndcondX := vNAND2(vAND3(vNOT(BndUopMuxUopType_0SchB[2]),
!                         vNOT(BndUopMuxUopType_0SchB[1]),
!                         BndUopMuxUopType_0SchB[0]),
!                   vAND3(vNOT(BndUopMuxUopType_1SchB[2]),
!                         vNOT(BndUopMuxUopType_1SchB[1]),
!                         BndUopMuxUopType_1SchB[0]));

! no smc alignment problems
! smc := (m2-op) and (store) and (eof) and (next instruction crosses a line)
!bndcond6 := vNAND4(vAND3(vNOT(BndUopMuxUopType_0SchB[2]),
!                         BndUopMuxUopType_0SchB[1],
!                         vNOT(BndUopMuxUopType_0SchB[0])),
!                   BndUopMuxSubType_0SchB[0],
!                   vNOR3(vCMP(BndUopMuxFlowMarker_0SchB, P7U_NO_MARKER),
!                         vCMP(BndUopMuxFlowMarker_0SchB, P7U_END_FLOW),
!                         vCMP(BndUopMuxFlowMarker_0SchB, P7U_OIW_NOW)),
!                   BndUopMuxLineCross_1SchB);

! fixme: Move the following rule to ucode:
! Ucode must mark I1 and B syllables as never bundle with successor
! 1st uop is not i1
!bndcondX := vNAND2(BndUopMuxUopType_0SchB[2],
!                   BndUopMuxUopType_0SchB[0]);


! Allow ucode to override normal dependency checking
BndForceDualSchB := vAND2(vNOT(BndUopMuxBndHint_0SchB[1]),
                          BndUopMuxBndHint_0SchB[0]);
BndCtlDependSchB := vOR2(vNAND4(bndcond0, bndcond1, bndcond2, bndcond3),
                         vNAND3(bndcond4, bndcond5, bndcond7));

END "bnddeps MAIN";
END "bnddeps";
```

```
!=========================================================================
!&!     $Header: /tmp/rtl/fpucntrls.hdl,v 1.22 1998/12/04 00:11:53 gwelte Exp $
!=========================================================================
!
! fpucntrls.hdl
!
! Author:            Rohit Bhatia
! Date:              09/29/97
!
! Copyright (c) 1997 Hewlett-Packard
! HP Proprietary
!
! RCS information:
!       $Author: gwelte $
!       $Date: 1998/12/04 00:11:53 $
!       $Revision: 1.22 $
!       $State: Exp $
!       $Locker:  $
!
!-----------------------------------------------------------------------
! fpucntrls Fub functions
!=======================================================================

!FUB <fubname> @<attributes>;
FUB fpucntrls;

BEGIN  "fpucntrls"

!=======================================================================
!                      Global Constant/Macro Defines
!=======================================================================

SOURCEFILE "p7includes.def"
SOURCEFILE "syn_maclib.def"
SOURCEFILE "dp_maclib.def"
SOURCEFILE "emparams.def"
SOURCEFILE "p7params.def"
SOURCEFILE "emdecode.def"
SOURCEFILE "reset.def"
SOURCEFILE "fpu.def"

!!!=======================================================================
!!!                      Local Constant/Macro Defines
!!!=======================================================================

!SOURCEFILE "<fubname>.def" <optional>
!<OR>
!<local macros> <optional>

DEFINE  Dsr1_RdFp4H     = [FpuDsr1RdFp4H]
DEFINE  Dsr1_WrFp4H     = [FpuDsr1WrFp4H]
DEFINE  Dsr2_RdFp4H     = [FpuDsr2RdFp4H]
DEFINE  Dsr2_WrFp4H     = [FpuDsr2WrFp4H]

CTINTEGER k,p,sfnum ENDC

!!!=======================================================================
!!!                      Interface/Clock Declaration
!!!=======================================================================

SOURCEFILE "fpucntrls.ifc"

!!!=======================================================================
!!!                        Local Node/Variable Declarations
!!!==============================================================================

SOURCEFILE "fpucntrls.int"

!!!==============================================================================
!!!                      Instrumentation and Debug Variables
!!!==============================================================================

!SOURCEFILE "<fubname>.inst" <optional>
!<OR>
!<local instrumentation variable declaration,
! WHENSTABLE PROC,
! Instrumentation procedures> <optional>

!!!==============================================================================
!!!                              Clock Description
!!!==============================================================================

SOURCEFILE "clock.typ"

!!!==============================================================================
!!!                              MAIN Procedure
!!!==============================================================================

!IFC P7_IS_FV THENC SOURCEFILE "<fubname>.dir" ENDC

STRUCTURAL MAIN;

BEGIN "fpucntrls MAIN"


!!!==============================================================================
!!! PSR info
!!!==============================================================================
vDFF(CLKFPU4,DcsPsrDFH111H,FpuPsrDFHFp1H);
vDFF(CLKFPU4,DcsPsrDFL111H,FpuPsrDFLFp1H);

!!!==============================================================================
!!! SIR Stall Generation
!!!==============================================================================

FORC k := 0 UPTO 1 DOC

vEDFF(CLKFPU4, FpuDsr1WrFp1H, f%k%_UnsafeFp1H, f%k%_UnsafeRecFp2H);
f%k%_UnsafeMxFp1H          :=
        vMUXE2(f%k%_UnsafeFp1H, f%k%_UnsafeRecFp2H,
                FpuDsr1RdFp1H);
ENDC


FpuDisFmaSIRFp1H           :=
        vOR2(FpuF0FmaTypeFp1H AND fp_DisableSIR111H AND FpuF0VldOpFp1H,
             FpuF1FmaTypeFp1H AND fp_DisableSIR111H AND FpuF1VldOpFp1H);

FpuEarlyGenSirFp1H         :=
        vOR3(FpuF0EarlyFltFp1H,FpuF1EarlyFltFp1H,
             FpuDisFmaSIRFp1H);
FpuGenSirStallFp1H         :=
        vOR3((f0_UnsafeMxFp1H) AND FpuF0UnsafeVldFp1H,
             (f1_UnsafeMxFp1H) AND FpuF1UnsafeVldFp1H,
             FpuEarlyGenSirFp1H);
```

```
vRESDFF(CLKFPU4,
        XpnBruFlushWrbH OR                                          ! reset
        (SpuStallExeH AND NOT DccStallDetH) OR
        fp_SirStallFp2H,
        NOT DccStallDetH,                                           ! enable
        FpuGenSirStallFp1H,FpuGenSirStallFp2H);                     ! d, q

vRSDFF(CLKFPU4,
        XpnBruFlushWrbH,                                            ! reset
        FpuGenSirStallFp2H,FpuGenSirStallFp3H);                     ! d, q
vDFF(CLKFPU4,FpuGenSirStallFp3H,FpuGenSirStallFp4H);

FpuSirStallFp2H :=
        vOR3(FpuGenSirStallFp2H,
             FpuGenSirStallFp3H,
             FpuGenSirStallFp4H);

! Local copy of FPU internal consumers
fp_SirStallFp2H := dpBUFFER(FpuSirStallFp2H);

!!!==========================================================================
!!! REN
!!!==========================================================================
FORC k := 0 UPTO 1 DOC
  ASSIGNC p := k + 4 ENDC

! Port Valid
vRESDFF(CLKFPU4,FpuFlushRenH,FpuEnableRenH,
        IsdPort%p%VldRenH,FpuF%k%SylVldRegH);
vRESDFF(CLKFPU4,FpuFlushRenH,FpuEnableRenH,
        FpuF%k%HasPred1RenH,FpuF%k%HasPred1RegH);
vRESDFF(CLKFPU4,FpuFlushRenH,FpuEnableRenH,
        FpuF%k%HasPred2RenH,FpuF%k%HasPred2RegH);
ENDC

! Simd op can only be on one port
FV_MUTEX(RendF0SimdOpRenH, RendF1SimdOpRenH,
         ("FP-SIMD op on both F0 and F1"));

! Replicate syllables on both ports for a simd op
FpuPort4IveBitRenB      :=
        vMUXE2(IsdPort4IveBitRenB, IsdPort5IveBitRenB,
               RendF1SimdOpRenH);
FpuPort4SyllRenH[40:6]  :=
        vMUXE2(RenPort4SyllRenH[40:6], RenPort5SyllRenH[40:6],
               RendF1SimdOpRenH);

FpuPort5IveBitRenB      :=
        vMUXE2(IsdPort5IveBitRenB, IsdPort4IveBitRenB,
               RendF0SimdOpRenH);
FpuPort5SyllRenH[40:6]  :=
        vMUXE2(RenPort5SyllRenH[40:6], RenPort4SyllRenH[40:6],
               RendF0SimdOpRenH);

! Replicate src ids on both ports for a simd op
FORC k := 0 UPTO 2 DOC
FpuPort4pSrc%k%RenH     :=
        vMUXE2(RenPort4pSrc%k%RenH, RenPort5pSrc%k%RenH,
               RendF1SimdOpRenH);
FpuPort5pSrc%k%RenH     :=
        vMUXE2(RenPort5pSrc%k%RenH, RenPort4pSrc%k%RenH,
               RendF0SimdOpRenH);
ENDC
```

```
vRESDFF(CLKFPU4,FpuFlushRenH,FpuEnableRenH,
        RendF0SimdOpRenH OR RendF1SimdOpRenH, f0_SimdOpRegH);
vRESDFF(CLKFPU4,FpuFlushRenH,FpuEnableRenH,
        RendF0SimdOpRenH OR RendF1SimdOpRenH, f1_SimdOpRegH);
!!!=============================================================================
!!! REG
!!!=============================================================================

! Check for WAW
FpuDestSameRegH          :=
        vCMP(FpuF0DestIdRegH, FpuF1DestIdRegH) AND
        FpuF0GoodDestRegH AND FpuF1GoodDestRegH;

FORC k := 0 UPTO 1 DOC

!! Mac start signals
f%k%_MacStartRegL        :=
        vOR2(FpuF%k%SylVldRegH,f%k%_SimdOpRegH);

f%k%_SimdMacStartRegL    := FpuF%k%SylVldRegH;

FpuF%k%GoodDestRegH      :=
        vAND3(FpuF%k%SylVldRegH, FpuF%k%HasDestRegH,
              vBOR(FpuF%k%DestIdRegH[6:1]));

!! Illegal Op Flt
FpuF%k%IllegalOpRegH     :=
        vOR3(FpuF%k%RsvdInstRegH,
             FpuF%k%IllgPredRegH,
             FpuF%k%HasDestRegH AND (vBNOR(FpuF%k%DestIdRegH[6:1])));

vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%IllegalOpRegH,FpuF%k%IllegalOpFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%IllgPredUncRegH,FpuF%k%IllgPredUncFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%BreakInstRegH,FpuF%k%BreakInstFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%NopRegH,FpuF%k%NopFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%AccFRFHiRegH,FpuF%k%AccFRFHiFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%AccFRFLoRegH,FpuF%k%AccFRFLoFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%ModHiRegH,FpuF%k%ModHiFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%ModLoRegH,FpuF%k%ModLoFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%DivTypeRegH,FpuF%k%DivTypeFp1H);      ! F6+F7
vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%CmpTypeRegH,FpuF%k%CmpTypeFp1H);      ! F4+F5
vEDFF(CLKFPU4,FpuEnableRegH,f%k%_CmpClsTypeRegH[9],FpuF%k%CmpClsUncFp1H); ! Unc type

vEDFF(CLKFPU4,FpuEnableRegH,FpuF%k%DestIdRegH,FpuF%k%DestIdFp1H);

! Port,Dst Valid, Predicate Vld
vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,
        FpuF%k%SylVldRegH,FpuF%k%SylVldFp1H);
vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,
        FpuF%k%FlgTypeRegH,FpuF%k%UnsafeVldFp1H);
vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,
        FpuF%k%GoodDestRegH,FpuF%k%GoodDestFp1H);
vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,
        FpuF%k%HasPred1RegH,FpuF%k%HasPred1Fp1H);
vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,
        FpuF%k%HasPred2RegH,FpuF%k%HasPred2Fp1H);
ENDC

vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,f0_SimdOpRegH,FpuSimdResFp1H);
vEDFF(CLKFPU4,FpuEnableRegH,FpuDestSameRegH,FpuDestSameFp1H);

!!!=============================================================================
!!! FP1
!!!=============================================================================

FORC k := 0 UPTO 1 DOC
  ASSIGNC p := k + 4 ENDC

!!--------------
! Port valid qualified with Qp -> VldOp
FpuF%k%VldOpFp1H        := FpuF%k%SylVldFp1H AND SpuQp%p%ExeH;

  IFC (k = 0) THENC
f%k%_VldFinstFp1H       :=
        vAND3(FpuF%k%GoodDestFp1H AND FpuF%k%SylVldFp1H,
              FpuDestSameFp1H NAND SpuQp5ExeH,
              SpuQp%p%ExeH);
FpuF%k%FRFUpdFp1H       :=
        vAND3(FpuF%k%GoodDestFp1H AND FpuF%k%SylVldFp1H,
              FpuDestSameFp1H NAND SpuQp5ExeH,
              SpuQp%p%ExeH);
FpuF%k%VldBypFp1H       :=
        vAND3(FpuF%k%GoodDestFp1H AND FpuF%k%SylVldFp1H,
              FpuDestSameFp1H NAND SpuQp5ExeH,
              SpuQp%p%ExeH);
  ENDC
  IFC (k = 1) THENC
f%k%_VldFinstFp1H       :=
        vAND3(FpuF%k%GoodDestFp1H AND FpuF%k%SylVldFp1H,
              FpuDestSameFp1H NAND SpuQp4ExeH,
              SpuQp%p%ExeH);
FpuF%k%FRFUpdFp1H       :=
        vAND3(FpuF%k%GoodDestFp1H AND FpuF%k%SylVldFp1H,
              FpuDestSameFp1H NAND SpuQp4ExeH,
              SpuQp%p%ExeH);
FpuF%k%VldBypFp1H       :=
        vAND3(FpuF%k%GoodDestFp1H AND FpuF%k%SylVldFp1H,
              FpuDestSameFp1H NAND SpuQp4ExeH,
              SpuQp%p%ExeH);
  ENDC
!!--------------

!!--------------
! Generate the early faults
FpuF%k%IllglOpFltFp1H   :=                      ! Illegal Op Flt
        vOR2(FpuF%k%IllegalOpFp1H AND FpuF%k%VldOpFp1H,
             FpuF%k%IllgPredUncFp1H AND FpuF%k%SylVldFp1H);

FpuF%k%BreakFltFp1H     :=                      ! Break Flt
        vAND2(FpuF%k%BreakInstFp1H, FpuF%k%VldOpFp1H);

FpuF%k%DisFRFHiFltFp1H  :=                      ! Disabled FP Reg Hi Flt
        vAND3(FpuF%k%AccFRFHiFp1H, FpuPsrDFHFp1H, FpuF%k%VldOpFp1H);
FpuF%k%DisFRFLoFltFp1H  :=                      ! Disabled FP Reg Lo Flt
        vAND3(FpuF%k%AccFRFLoFp1H, FpuPsrDFLFp1H, FpuF%k%VldOpFp1H);

FpuF%k%ResFldFltFp1H    :=
        vAND2(FpuF%k%RsvdFldFp1H, FpuF%k%VldOpFp1H);


FpuF%k%EarlyFltFp1H     :=
        vOR6(FpuF%k%IllglOpFltFp1H,
             FpuF%k%BreakFltFp1H,
             FpuF%k%DisFRFHiFltFp1H,
             FpuF%k%DisFRFLoFltFp1H,
            FpuF%k%ResFldFltFp1H,
            FpuF%k%SpecOpFltFp1H);
!!--------------

!!--------------
! Check if unsafe is signalled for PMU FalseSIR event
FpuF%k%IsUnsafeExeH     :=      ! Unsafe_signalled & Vld_Arith_Inst
        vAND3(f%k%_UnsafeMxFp1H, FpuF%k%UnsafeVldFp1H, Pmc8f%k%TagExeH);
!!--------------

!!--------------
! Generate dest modify signals for PSR{mfh,mfl}
FpuF%k%ModHiExeH        :=
        vAND2(FpuF%k%ModHiFp1H, f%k%_VldFinstFp1H);
FpuF%k%ModLoExeH        :=
        vAND2(FpuF%k%ModLoFp1H, f%k%_VldFinstFp1H);
!!--------------

!!--------------
! Pred valids
FpuF%k%Pred0VldFp1H     :=
        vOR3(vAND4(FpuF%k%HasPred1Fp1H,
                   FpuF%k%CmpTypeFp1H,NOT FpuF%k%CmpClsUncFp1H,SpuQp%p%ExeH),
             vAND3(FpuF%k%HasPred1Fp1H,
                   FpuF%k%CmpTypeFp1H, FpuF%k%CmpClsUncFp1H),
             vAND2(FpuF%k%HasPred1Fp1H,FpuF%k%DivTypeFp1H));
FpuF%k%Pred1VldFp1H     :=
        vOR3(vAND4(FpuF%k%HasPred2Fp1H,
                   FpuF%k%CmpTypeFp1H,NOT FpuF%k%CmpClsUncFp1H,SpuQp%p%ExeH),
             vAND3(FpuF%k%HasPred2Fp1H,
                   FpuF%k%CmpTypeFp1H, FpuF%k%CmpClsUncFp1H),
             vAND2(FpuF%k%HasPred2Fp1H,FpuF%k%DivTypeFp1H));
!!--------------

! Valid bits
vRESDFF(CLKFPU4,FpuFlushFp1H,FpuAdvFp1H,FpuF%k%SylVldFp1H,FpuF%k%SylVldFp2H);
vRESDFF(CLKFPU4,FpuFlushFp1H,FpuAdvFp1H,FpuF%k%VldOpFp1H,FpuF%k%VldOpFp2H);
!vRESDFF(CLKFPU4,FpuFlushFp1H,FpuAdvFp1H,f%k%_VldFinstFp1H,f%k%_VldFinstFp2H);

vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%UnsafeVldFp1H, FpuF%k%UnsafeVldFp2H);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%IsUnsafeExeH, FpuF%k%IsUnsafeDetH);

vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%VldBypFp1H,FpuF%k%VldBypFp2H);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%Pred0VldFp1H,FpuPred0Vld%p%Fp2H);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%Pred1VldFp1H,FpuPred1Vld%p%Fp2H);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%FRFUpdFp1H,FpuF%k%FRFUpdFp2H);

vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%ModHiExeH,FpuF%k%ModHiDetH);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%ModLoExeH,FpuF%k%ModLoDetH);

vEDFF(CLKFPU4,FpuEnableExeH,
        FpuF%k%FlopCntExeH AND Pmc8f%k%TagExeH::3, FpuF%k%FlopCntDetH);

! Propagate the early faults
vEDFF(CLKFPU4,FpuEnableExeH,FpuF%k%IllglOpFltFp1H,FpuF%k%IllglOpFltFp2H);
vEDFF(CLKFPU4,FpuEnableExeH,FpuF%k%BreakFltFp1H,FpuF%k%BreakFltFp2H);
vEDFF(CLKFPU4,FpuEnableExeH,FpuF%k%DisFRFHiFltFp1H,FpuF%k%DisFRFHiFltFp2H);
vEDFF(CLKFPU4,FpuEnableExeH,FpuF%k%DisFRFLoFltFp1H,FpuF%k%DisFRFLoFltFp2H);
vEDFF(CLKFPU4,FpuEnableExeH,FpuF%k%ResFldFltFp1H,FpuF%k%ResFldFltFp2H);
vEDFF(CLKFPU4,FpuEnableExeH,FpuF%k%SpecOpFltFp1H,FpuF%k%SpecOpFltFp2H);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuF%k%EarlyFltFp1H,FpuF%k%EarlyFltFp2H);
ENDC

! SimdResult signal piped down for bpx
vDFF(CLKFPU4,FpuSimdResFp1H,FpuSimdResFp2H);


!!!=================================================================
!!! FP2
!!!=================================================================

FORC k := 0 UPTO 1 DOC
  ASSIGNC p := k + 4 ENDC

vDFF(CLKFPU4,FpuF%k%FlopCntDetH,FpuF%k%FlopCntWrbH);
vRSDFF(CLKFPU4,FpuFlushFp2H,FpuF%k%EarlyFltFp2H,FpuF%k%EarlyFltFp3H);
vRSDFF(CLKFPU4,FpuFlushDetH,FpuF%k%IsUnsafeDetH,FpuF%k%IsUnsafeWrbH);
! Valid bits
vRSDFF(CLKFPU4,FpuFlushFp2H,FpuF%k%SylVldFp2H,FpuF%k%SylVldFp3H);
vRSDFF(CLKFPU4,FpuFlushFp2H,FpuF%k%VldOpFp2H,FpuF%k%VldOpFp3H);
!vRSDFF(CLKFPU4,FpuFlushFp2H,f%k%_VldFinstFp2H,f%k%_VldFinstFp3H);
vRSDFF(CLKFPU4,FpuFlushDetH,FpuF%k%VldBypFp2H,FpuF%k%VldBypFp3H);
vRSDFF(CLKFPU4,FpuFlushDetH,FpuF%k%FRFUpdFp2H,FpuF%k%FRFUpdFp3H);
ENDC

! SimdResult signal piped down for bpx
vDFF(CLKFPU4,FpuSimdResFp2H,FpuSimdResFp3H);

!!!=================================================================
!!! FP3
!!!=================================================================

FORC k := 0 UPTO 1 DOC
  ASSIGNC p := k + 4 ENDC

! Qualify valids with XpnCommits
FpuF%k%SylVldCmtFp3H    := FpuF%k%SylVldFp3H AND XpnCommit%p%WrbB;
FpuF%k%VldOpCmtFp3H     := FpuF%k%VldOpFp3H AND XpnCommit%p%WrbB;
FpuF%k%VldBypCmtFp3H    := FpuF%k%VldBypFp3H AND XpnCommit%p%WrbB;

vDFF(CLKFPU4,FpuF%k%FlopCntWrbH AND FpuF%k%VldOpCmtFp3H::3, FpuF%k%FlopCntFp4H);
vDFF(CLKFPU4,FpuF%k%IsUnsafeWrbH, FpuF%k%IsUnsafeFd4H);
! Valid bits
vDFF(CLKFPU4,FpuF%k%SylVldCmtFp3H,FpuF%k%SylVldCmtFp4H);
vDFF(CLKFPU4,FpuF%k%VldOpCmtFp3H,FpuF%k%VldOpCmtFp4H);
!vDFF(CLKFPU4,f%k%_VldFinstFp3H AND XpnCommit%p%WrbB,f%k%_VldFinstFp4H);
vDFF(CLKFPU4,FpuF%k%FRFUpdFp3H AND XpnCommit%p%WrbB,FpuF%k%FRFUpdFp4H);
vDFF(CLKFPU4,FpuF%k%EarlyFltFp3H AND XpnCommit%p%WrbB, FpuF%k%EarlyFltIntFp4H);
vDFF(CLKFPU4,f%k%_SimdOpFp3H, f%k%_SimdOpFp4H);
ENDC

!!!=================================================================
!!! FP4
!!!=================================================================

FV_FORBIDDEN(FpuF0FlopCntFp4H[2] AND FpuF1FlopCntFp4H[2],
        ("Both F0 and F1 have a 4-flop instruction"));
```

```
FpuFlopCntFp4H[2:0]       :=
        dpADD(FpuF0FlopCntFp4H[2:0], FpuF1FlopCntFp4H[2:0]);

FORC k := 0 UPTO 1 DOC
! mac1 operates on simdlo
f%k%_SimdMacINVTrapFp4H := vAND2(f%k%_SimdOpFp4H, f1_MacINVTrapFp4H);
f%k%_SimdOVFTrapFp4H    := vAND2(f%k%_SimdOpFp4H, f1_OVFTrapFp4H);
f%k%_SimdUDFTrapFp4H    := vAND2(f%k%_SimdOpFp4H, f1_UDFTrapFp4H);
f%k%_SimdINXTrapFp4H    := vAND2(f%k%_SimdOpFp4H, f1_INXTrapFp4H);
f%k%_SimdFPAFp4H        := vAND2(f%k%_SimdOpFp4H, f1_FPAFp4H);

f%k%_TrapStatFp4H[7:0]  :=
        f%k%_FPAFp4H &
        f%k%_SWAFp4H &
        (f%k%_INVTrapFp4H OR f%k%_MacINVTrapFp4H) &
        f%k%_DENTrapFp4H &
        f%k%_DIVTrapFp4H &
        f%k%_OVFTrapFp4H &
        f%k%_UDFTrapFp4H &
        f%k%_INXTrapFp4H ;
f%k%_SimdTrapStatFp4H[7:0]      :=
        f%k%_SimdFPAFp4H &
        f%k%_SimdSWAFp4H &
        (f%k%_SimdINVTrapFp4H OR f%k%_SimdMacINVTrapFp4H) &
        f%k%_SimdDENTrapFp4H &
        f%k%_SimdDIVTrapFp4H &
        f%k%_SimdOVFTrapFp4H &
        f%k%_SimdUDFTrapFp4H &
        f%k%_SimdINXTrapFp4H ;


!!--------------
! Normal FP Trap DSR
FP_2STG_DSR(CLKFPU4, f%k%_TrapStatFp4H, f%k%_TrapStatFd4H,
        f%k%Dsr1_TrapStatFd4H,
        f%k%Dsr2_TrapStatFp4H, f%k%Dsr2_TrapStatFd4H,
        f%k%DsrA_TrapStatFd4H);

! SIMD FP Trap DSR
FP_2STG_DSR(CLKFPU4, f%k%_SimdTrapStatFp4H, f%k%_SimdTrapStatFd4H,
        f%k%Dsr1_SimdTrapStatFd4H,
        f%k%Dsr2_SimdTrapStatFp4H, f%k%Dsr2_SimdTrapStatFd4H,
        f%k%DsrA_SimdTrapStatFd4H);

! EBC DSR
FP_2STG_DSR(CLKFPU4, f%k%_EBCFp4H, f%k%_EBCFd4H,
        f%k%Dsr1_EBCFd4H,
        f%k%Dsr2_EBCFp4H, f%k%Dsr2_EBCFd4H,
        f%k%DsrA_EBCFd4H);
!!--------------


FpuF%k%FaultFd4H          :=
        vOR2(vBOR(f%k%_TrapStatFd4H[6:3]),
             vBOR(f%k%_SimdTrapStatFd4H[6:3]));
FpuF%k%TrapFd4H :=
        vOR2(vBOR(f%k%_TrapStatFd4H[2:0]),
             vBOR(f%k%_SimdTrapStatFd4H[2:0]));

! Drive FP Exceptions info to XPN
FpuF%k%ExcpFltFp4H        :=
        vAND3(FpuExcpUpdFp4H, FpuF%k%FaultFd4H, FpuF%k%UnsafeVldFp2H);
FpuF%k%ExcpTrpFp4H        :=
        vAND3(FpuExcpUpdFp4H, FpuF%k%TrapFd4H, FpuF%k%UnsafeVldFp2H);

vEDFF(CLKFPU4, FpuExcpAdvExeH, FpuF%k%ExcpFltFp4H, FpuF%k%ExcpFltDetH); ! FP fault
vEDFF(CLKFPU4, FpuExcpAdvExeH, FpuF%k%ExcpTrpFp4H, FpuF%k%ExcpTrpDetH); ! FP trap

vEDFF(CLKFPU4, FpuExcpAdvExeH, f%k%_TrapStatFd4H, f%k%_TrapStatDetH);
vEDFF(CLKFPU4, FpuExcpAdvExeH, f%k%_SimdTrapStatFd4H, f%k%_SimdTrapStatDetH);
vEDFF(CLKFPU4, FpuExcpAdvExeH, f%k%_EBCFd4H, f%k%_EBCDetH); ! EBC
!

FpuF%k%IsrCodeDetH[0]   :=
        vMUXE4(f%k%_SimdTrapStatDetH[2],                        ! EM  FpTrap
               f%k%_TrapStatDetH[5],                            ! EM  FpFault
               f%k%_TrapStatDetH[5] OR f%k%_SimdTrapStatDetH[5],! Bif FpTrap
               f%k%_TrapStatDetH[5] OR f%k%_SimdTrapStatDetH[5],! Bif FpFault
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[1]   :=
        vMUXE4(f%k%_SimdTrapStatDetH[1],
               f%k%_TrapStatDetH[4],
               f%k%_TrapStatDetH[4] OR f%k%_SimdTrapStatDetH[4],
               f%k%_TrapStatDetH[4] OR f%k%_SimdTrapStatDetH[4],
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[2]   :=
        vMUXE4(f%k%_SimdTrapStatDetH[0],
               f%k%_TrapStatDetH[3],
               f%k%_TrapStatDetH[3] OR f%k%_SimdTrapStatDetH[3],
               f%k%_TrapStatDetH[3] OR f%k%_SimdTrapStatDetH[3],
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[3]   :=
        vMUXE4(f%k%_SimdTrapStatDetH[7],
               f%k%_TrapStatDetH[6],
               f%k%_TrapStatDetH[2] OR f%k%_SimdTrapStatDetH[2],
               f%k%_TrapStatDetH[2] OR f%k%_SimdTrapStatDetH[2],
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[4]   :=
        vMUXE4(f%k%_TrapStatDetH[2],
               f%k%_SimdTrapStatDetH[5],
               f%k%_TrapStatDetH[1] OR f%k%_SimdTrapStatDetH[1],
               f%k%_TrapStatDetH[1] OR f%k%_SimdTrapStatDetH[1],
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[5]   :=
        vMUXE4(f%k%_TrapStatDetH[1],
               f%k%_SimdTrapStatDetH[4],
               f%k%_TrapStatDetH[0] OR f%k%_SimdTrapStatDetH[0],
               f%k%_TrapStatDetH[0] OR f%k%_SimdTrapStatDetH[0],
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[6]   :=
        vMUXE4(f%k%_TrapStatDetH[0],
               f%k%_SimdTrapStatDetH[3],
               'b0 OR 'b0,
               'b0 OR 'b0,
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[7]   :=
        vMUXE4(f%k%_TrapStatDetH[7],
               f%k%_SimdTrapStatDetH[6],
               f%k%_TrapStatDetH[6] OR f%k%_SimdTrapStatDetH[6],
               f%k%_TrapStatDetH[6] OR f%k%_SimdTrapStatDetH[6],
                FpuIveMode111H & FpuF%k%ExcpFltDetH);
FpuF%k%IsrCodeDetH[8]   :=
        vMUXE4(f%k%_EBCDetH,
               'b0,
               FpuBifMMX2OpFp2H,
               FpuBifMMX2OpFp2H,
                FpuIveMode111H & FpuF%k%ExcpFltDetH);

ENDC

FpuIsrCodeDetH[8:0]      :=
        vMUXE4(FpuF1IsrCodeDetH[8:0],
               FpuF0IsrCodeDetH[8:0],
               FpuF0IsrCodeDetH[8:0] OR FpuF1IsrCodeDetH[8:0],
               FpuF0IsrCodeDetH[8:0] OR FpuF1IsrCodeDetH[8:0],
                FpuBifMMX2OpFp2H & (FpuF0ExcpFltDetH OR FpuF0ExcpTrpDetH));

FORC k := 0 UPTO 1 DOC
! Valid bits
vDFF(CLKFPU4,FpuF%k%SylVldCmtFp4H,FpuF%k%SylVldCmtFwbH);
vDFF(CLKFPU4,FpuF%k%VldOpCmtFp4H,FpuF%k%VldOpCmtFwbH);
vDFF(CLKFPU4,'b0,f%k%_VldFinstWrbH);
vDFF(CLKFPU4,FpuF%k%FRFUpdFp4H,FpuF%k%FRFUpdFwbH);
ENDC

! Drive Early Fault indications to XPN
FpuF0EarlyFltFp4H        :=
        vAND2(FpuExcpUpdFp4H, FpuF0EarlyFltFp2H) ;
FpuF1EarlyFltFp4H        :=
        vAND2(FpuExcpUpdFp4H, FpuF1EarlyFltFp2H) ;
FpuIllglOpFltFp4H        :=
        vOR2(vAND2(FpuF0IllglOpFltFp2H, FpuF0EarlyFltFp4H),
             vAND3(FpuF1IllglOpFltFp2H, FpuF1EarlyFltFp4H,
                                    NOT FpuF0EarlyFltFp4H));
FpuBreakFltFp4H :=
        vOR2(vAND2(FpuF0BreakFltFp2H,   FpuF0EarlyFltFp4H),
             vAND3(FpuF1BreakFltFp2H,   FpuF1EarlyFltFp4H,
                                    NOT FpuF0EarlyFltFp4H));
FpuDisFRFHiFltFp4H       :=
        vOR2(vAND2(FpuF0DisFRFHiFltFp2H,FpuF0EarlyFltFp4H),
             vAND3(FpuF1DisFRFHiFltFp2H,FpuF1EarlyFltFp4H,
                                    NOT FpuF0EarlyFltFp4H));
FpuDisFRFLoFltFp4H       :=
        vOR2(vAND2(FpuF0DisFRFLoFltFp2H,FpuF0EarlyFltFp4H),
             vAND3(FpuF1DisFRFLoFltFp2H,FpuF1EarlyFltFp4H,
                                    NOT FpuF0EarlyFltFp4H));
FpuResFldFltFp4H         :=
        vOR2(vAND2(FpuF0ResFldFltFp2H,  FpuF0EarlyFltFp4H),
             vAND3(FpuF1ResFldFltFp2H,  FpuF1EarlyFltFp4H,
                                    NOT FpuF0EarlyFltFp4H));
FpuSpecOpFltFp4H         :=
        vOR2(vAND2(FpuF0SpecOpFltFp2H,  FpuF0EarlyFltFp4H),
             vAND3(FpuF1SpecOpFltFp2H,  FpuF1EarlyFltFp4H,
                                    NOT FpuF0EarlyFltFp4H));

!!!=========================================================================
!!! FWB
!!!=========================================================================

vDFF(CLKFPU4, FpuIsrCodeDetH, FpuIsrCodeWrbH);
FpuIsrCodeFwbH[8:0]      := dpBUFFER(FpuIsrCodeWrbH[8:0]);

!!!=========================================================================
!!! Deferred Stall Register(DSR) Logic
!!!=========================================================================

!!--------------
! Generate the update and drive DSR signals
```

```
vDFF(CLKFPU4, SpuStallExeH, FpuExeStallDetH);

FpuDetStallDetH         := vOR2(DccStallDetH, fp_SirStallFp2H);
vDFF(CLKFPU4, FpuDetStallDetH, FpuDetStallWrbH);
vDFF(CLKFPU4, fp_SirStallFp2H, fp_SirStallWrbH);

FpuDsr2WrFp2H   :=                                     ! Load the DetDSR
        vAND2(FpuDetStallDetH, NOT FpuDetStallWrbH);
vDFF(CLKFPU4, FpuDsr2WrFp2H, FpuDsr2WrFp3H);
vDFF(CLKFPU4, FpuDsr2WrFp3H, FpuDsr2WrFp4H);

FpuDsr2RdFp2H   :=                                     ! Read the DetDSR
        vAND2(NOT FpuDetStallDetH, FpuDetStallWrbH);
vDFF(CLKFPU4, FpuDsr2RdFp2H, FpuDsr2RdFp3H);
vDFF(CLKFPU4, FpuDsr2RdFp3H, FpuDsr2RdFp4H);

FpuDsr1WrFp1H   :=                                     ! Load the ExeDSR
        vAND3(FpuDetStallDetH, NOT SpuStallExeH, NOT FpuDsr1Vld111H);
vDFF(CLKFPU4, FpuDsr1WrFp1H, FpuDsr1WrFp2H);
vDFF(CLKFPU4, FpuDsr1WrFp2H, FpuDsr1WrFp3H);
vDFF(CLKFPU4, FpuDsr1WrFp3H, FpuDsr1WrFp4H);

FpuDsr1VldFp1H  :=
        vOR2(FpuDsr1WrFp1H, vAND2(FpuDsr1Vld111H, NOT FpuAdvRegH));
vRSDFF(CLKFPU4, FpuResetAsynch111H, FpuDsr1VldFp1H, FpuDsr1Vld111H);


FpuDsr1RdFp1H   :=                                     ! Read the ExeDSR
        vAND2(FpuAdvRegH, FpuDsr1Vld111H);
vDFF(CLKFPU4, FpuDsr1RdFp1H, FpuDsr1RdFp2H);
vDFF(CLKFPU4, FpuDsr1RdFp2H, FpuDsr1RdFp3H);
vDFF(CLKFPU4, FpuDsr1RdFp3H, FpuDsr1RdFp4H);

FpuExcpUpdFp2H  :=
        vAND2(fp_SirStallFp2H, NOT fp_SirStallWrbH);
vDFF(CLKFPU4, FpuExcpUpdFp2H, FpuExcpUpdFp3H);
vDFF(CLKFPU4, FpuExcpUpdFp3H, FpuExcpUpdFp4H);
FpuExcpAdvExeH  :=
        vOR2(NOT DccStallDetH, FpuExcpUpdFp4H);

!!--------------

!!!=====================================================================
!!! FP PMU Events
!!!=====================================================================

! ports 4,5(f0,f1) instructions which would have been retired if predicate was
! true instead of being false. this event is qualified with tag generated by
! IBR0/1 and pmc8 in isddfts.
FORC k := 0 UPTO 1 DOC
        ASSIGNC p := k+4 ENDC

pmc8f%k%tagregh := isddbgp%p%tagregh[0];
vRESDFF(CLKFPU4,FpuFlushRegH,FpuEnableRegH,Pmc8f%k%TagRegH,Pmc8f%k%TagExeH);

FpuSyllQpOff%k%Fp1H     :=
        vAND3(FpuF%k%SylVldFp1H, NOT SpuQp%p%ExeH, Pmc8f%k%TagExeH);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuSyllQpOff%k%Fp1H,FpuSyllQpOff%k%DetH);

!ports 4,5(f0,f1) instructions which are nops and have true predicates. this
!event is qualified with tag generated by IBR0/1 and pmc8 in isddfts.
FpuNopsVld%k%Fp1H       :=
        vand3(FpuF%k%VldOpFp1H,FpuF%k%NopFp1H, Pmc8f%k%TagExeH);
vRESDFF(CLKFPU4,FpuFlushExeH,FpuEnableExeH,
        FpuNopsVld%k%Fp1H,FpuNopsVld%k%DetH);
ENDC

! Count False SIR assertions
!   SIR && Vld_Unsafe_Inst && !(Fault_or_Trap)
FpuPmFalseSIRFd4H       :=
        vOR2(vAND2(FpuF0IsUnsafeFd4H,
                   vOR2(vBNOR(f0_TrapStatFd4H[5:0]),
                        vBNOR(f0_SimdTrapStatFd4H[5:0]))),
             vAND2(FpuF1IsUnsafeFd4H,
                   vOR2(vBNOR(f1_TrapStatFd4H[5:0]),
                        vBNOR(f1_SimdTrapStatFd4H[5:0]))));
vDFF(CLKFPU4, FpuPmFalseSIRFd4H, FpuPmFalseSIRFwbH);

!FpuSIRMcaFd4H  :=
!       vOR2(vAND3(FpuF0VldOpCmtFp4H, NOT FpuF0IsUnsafeFd4H,
!                  vOR2(vBOR(f0_TrapStatFd4H[5:0]),
!                       vBOR(f0_SimdTrapStatFd4H[5:0]))),
!            vAND3(FpuF1VldOpCmtFp4H, NOT FpuF1IsUnsafeFd4H,
!                  vOR2(vBOR(f1_TrapStatFd4H[5:0]),
!                       vBOR(f1_SimdTrapStatFd4H[5:0]))));
!
!FV_FORBIDDEN(vAND2(NOT FpuF0IsUnsafeFd4H,
!                  vOR2(vBOR(f0_TrapStatFd4H),vBOR(f0_SimdTrapStatFd4H))),
!       ("F0 signals FP Fault/Trap without SIR"));
!FV_FORBIDDEN(vAND2(NOT FpuF1IsUnsafeFd4H,
!                  vOR2(vBOR(f1_TrapStatFd4H),vBOR(f1_SimdTrapStatFd4H))),
!       ("F1 signals FP Fault/Trap without SIR"));

! Count number of flops retired
vDFF(CLKFPU4, FpuFlopCntFp4H, FpuPmFlopCntFwbH);

! Count failed fchkf inst
vEDFF(CLKFPU4, FpuExcpAdvExeH, FpuSpecOpFltFp4H, FpuPmFailedFchkfDetH);

!!!=========================================================================
!!! Pipeline advance/flush signals
!!!=========================================================================

! Official Reset Macro
resInt(CLKFPU4, FpuResInL, FpuResOutL, FpuResetAsynch111H);

! Delayed version of XpnFlush
vDFF(CLKFPU4,XpnBruFlushWrbH,FpuXBFlushWb1H);
FV_FORBIDDEN(FpuXBFlushWb1H AND XpnCommit4WrbB,
             ("Xpncommit for F0 is not zero 1 clk after xpnflush"));
FV_FORBIDDEN(FpuXBFlushWb1H AND XpnCommit5WrbB,
             ("Xpncommit for F1 is not zero 1 clk after xpnflush"));

! Pipestage enable signals for Main Pipeline
!FpuEnableExpH  :=
!       vNOR4(RseStallRenH,SpuStallExeH,DccStallDetH,fp_SirStallFp2H);
FpuEnableRenH   :=
        vNOR3(SpuStallExeH,DccStallDetH,fp_SirStallFp2H);
FpuEnableRegH   :=
        vNOR3(SpuStallExeH,DccStallDetH,fp_SirStallFp2H);
FpuEnableExeH   :=
        vNOR2(DccStallDetH,fp_SirStallFp2H);

! Pipestage flush signals for Main Pipeline
FpuFlushRenH    :=
        vOR3(FpuResetAsynch111H, FpuXBFlushWb1H, RseStallRenH AND FpuEnableRenH);
FpuFlushRegH    :=
        vOR2(FpuResetAsynch111H, FpuXBFlushWb1H);
FpuFlushExeH    :=
        vOR3(FpuResetAsynch111H, FpuXBFlushWb1H, SpuStallExeH AND FpuEnableExeH);
FpuFlushDetH    :=
        vOR4(FpuResetAsynch111H, FpuXBFlushWb1H, DccStallDetH, fp_SIRStallFp2H);

! Pipestage enable signals for FP Pipeline
FpuEnableFp1H   :=
        vNOT(DccStallDetH);

! Pipestage flush signals for FP Pipeline
FpuFlushFp1H    :=
        vOR3(FpuResetAsynch111H, FpuXBFlushWb1H, SpuStallExeH AND FpuEnableFp1H);
FpuFlushFp2H    :=
        vOR3(FpuResetAsynch111H, FpuXBFlushWb1H, DccStallDetH);

! Pipestage advance signals
FpuAdvExpH := vNOR4(RseStallRenH,SpuStallExeH,DccStallDetH,fp_SirStallFp2H);
FpuAdvRenH := vNOR3(SpuStallExeH,DccStallDetH,fp_SirStallFp2H);
FpuAdvRegH := vNOR3(SpuStallExeH,DccStallDetH,fp_SirStallFp2H);
FpuAdvFp1H := vNOT(DccStallDetH);

FpuEnableOnSpuRegH :=
        (SpuStallExeH OR DccStallDetH OR Fp_SirStallFp2H) NAND
        (NOT SpuStallExeH);

END "fpucntrls MAIN";

END  "fpucntrls";

!=========================================================================
!                     Modification History
!=========================================================================
!
! Name:
! Date:
! ECO: <iF applicable>
! Bug Number(s): <iF applicable>
! Description:
!
! Name:
! Date:
! ECO: <iF applicable>
! Bug Number(s): <iF applicable>
! Description:
!
!===========================================================================
!
! Modification Log:
! $Log: fpucntrls.hdl,v $
! Revision 1.22  1998/12/04 00:11:53  gwelte
! Author: gwelte@eslint4.fc.hp.com (Gary Welte)
! gun0_419 update
!
! Revision 1.21.2.1  1998/11/19 23:34:18  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01706: BRANCH FROM 1.21.1.1: Implement FPU pmu events, cleanup fp ctl
!
! Revision 1.21  1998/10/30 22:09:08  mjl
! Author: mjl@hpesmjl.fc.hp.com (Michael J. Lee)
! gun0_414 update
```

```
!
! Revision 1.20.2.1  1998/10/28 17:33:42  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01599: BRANCH FROM 1.20.1.1: FPU bug fixes
!
! Revision 1.20  1998/10/21 17:53:35  gwelte
! Author: gwelte@eslint4.fc.hp.com (Gary Welte)
! gun0_412 update
!
! Revision 1.19.2.2  1998/10/20 15:15:06  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01553: Fix FPU bug 3305
!
! Revision 1.19  1998/10/07 05:35:10  mjl
! Author: mjl@mtlgv21.fc.hp.com (Michael J. Lee)
! gun0_408 update
!
! Revision 1.18.2.1  1998/10/06 14:09:19  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01489: BRANCH FROM 1.18.1.1: Fix bug 3351
!
! Revision 1.18  1998/09/23 14:49:35  gwelte
! Author: gwelte@eslint4 (Gary Welte)
! gun0_404 update
!
! Revision 1.17.2.3  1998/09/18 20:41:20  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01407: Implement SoftSIMD and remove simdmacs
!
! Revision 1.17  1998/08/28 20:37:19  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_400 update
!
! Revision 1.16.1.1  1998/08/27 15:05:58  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01331: Add 2-stage DSR in FPU
!
! Revision 1.16  1998/08/12 21:43:59  h1cad
! Author: h1cad@mtlgv03.fc.hp.com (Generic CAD user)
! gun0_338 update
!
! Revision 1.15.2.1  1998/08/11 15:30:10  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01229: BRANCH FROM 1.15.1.1: Fix bug 2390
!
! Revision 1.15  1998/08/07 03:32:43  h1cad
! Author: h1cad@mtlgv01.fc.hp.com (Generic CAD user)
! gun0_337 update
!
! Revision 1.13.2.1  1998/08/04 14:42:39  pjk
! Author: pjk@hpespjk.fc.hp.com (Patrick Knebel)
! PCO_01182: BRANCH FROM 1.13.1.1: re-order bifrost fpu events from fpu->xpn->bif
!
! Revision 1.13.1.1  1998/07/30 17:15:40  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01164: Switch FPU over to DccStallDetH; Bif FP cflags cleanup; fix bug 1932
!
! Revision 1.13  1998/07/29 04:01:39  h1cad
! Author: h1cad@mtlgv01.fc.hp.com (Generic CAD user)
! gun0_335 update
!
! Revision 1.12.1.1  1998/07/28 14:30:43  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01137: Clean up IEU decode fubs; Fix bug 2379
!
! Revision 1.12  1998/07/22 16:53:59  h1cad
! Author: h1cad@mtlgv01.fc.hp.com (Generic CAD user)
! gun0_333 update
!
! Revision 1.11.1.2  1998/07/17 17:37:36  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_01092: bug fixes; acr fixes; Bif FP enhancements
!
! Revision 1.11  1998/07/01 23:10:52  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_330 update
!
! Revision 1.10.1.2  1998/06/29 21:29:37  ravi
! Author: ravi@hpesravi.fc.hp.com (Ravi Koshy)
! PCO_00995: Signal cleanup and aligning the immed field
!
! Revision 1.10  1998/06/11 19:49:09  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_327 update
!
! Revision 1.9.2.1  1998/06/09 21:58:14  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00923: BRANCH FROM 1.9.1.1: Scuds is history !!
!
! Revision 1.9  1998/05/22 18:45:01  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_322 update
!
! Revision 1.8.2.1  1998/05/20 16:31:11  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00833: BRANCH FROM 1.8.1.1: Partial fix for bug 1539/1557
!
! Revision 1.8  1998/04/27 21:28:16  gwelte
! Author: gwelte@eslint4 (Gary Welte)
! gun0_316 update
!
! Revision 1.7.2.1  1998/04/22 03:06:58  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00689: Bifrost support for FSR updates, FP exceptions
!
! Revision 1.7  1998/03/30 21:54:16  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_310 update
!
! Revision 1.6.3.1  1998/03/26 22:13:08  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00578: Bifrost support in FPU
!
! Revision 1.6.2.1  1998/03/25 17:10:54  ravi
! Author: ravi@hpesravi.fc.hp.com (Ravi Koshy)
! PCO_00574: cleaning up the fpu wrapper
!
! Revision 1.6.1.1  1998/03/23 20:48:56  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00566: Fix bugs 856 932 954 1004
!
! Revision 1.6  1998/03/18 00:49:39  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_306 update
!
! Revision 1.5.2.1  1998/03/16 22:17:41  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00545: Clean up FP Ctl-Stack exception interface
!
! Revision 1.5  1998/02/20 22:46:59  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_300 update
!
! Revision 1.4.3.2  1998/02/19 18:39:29  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00459: IEU,FPU bug fixes; Globals changes for IEU
!
! Revision 1.4  1998/01/27 00:52:55  gwelte
! Author: gwelte@intgv03.fc.hp.com (Gary Welte)
! gun0_227 update
!
! Revision 1.3.1.1  1998/01/22 22:10:18  pjr
! Author: pjr@hpespjr.fc.hp.com (Preston Renstrom)
! PCO_00372: New register file and bypass valids
!
! Revision 1.3  1998/01/14 02:17:46  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! gun0_223 update
!
! Revision 1.2.2.1  1998/01/09 22:21:15  rxb
! Author: rxb@hpesrxb.fc.hp.com (Rohit Bhatia)
! PCO_00321: Update FPU-XPN interface
!
! Revision 1.2  1997/10/28 20:55:04  h1cad
! Author: h1cad@hpesgd01.fc.hp.com (Generic CAD user)
! ! -r? /p7/nohdr
!
! Revision 1.1.1.5  1997/10/23 04:07:08  ravi
! Author: ravi@hpesravi.fc.hp.com (Ravi Koshy)
! PCO_00180: fpu cntrl signals moved up 1 cycle; new opcocdes/encodings
!
!============================================================================
```